An SMT Approach to a Multiparty Economic Scheduling Problem

Shriphani Palakodety, Guha Jayachandran, Aditya Thakur

ONAI®

UC DAVIS
UNIVERSITY OF CALIFORNIA

Setting

- Marketplace
- Matching buyers / sellers
- Buyer constraints
- Seller constraints
- Matching must honor both

Setting

- Cryptocurrency network
- Offer and Request computation
- Payment through cryptocurrency
- Constraints specified on both sides
- Match Requester and Offerer

Setting

- Rich specification:
  - Combine requests and offers
    - Offer algorithm X only if algorithm Y is provided (for example)
    - Price constraints on what you can offer/buy
    - Time constraints like timeouts, schedule-after-x etc.
- Rich operator algebra

Goals

- Match requests and offers
- Satisfy constraints
- Optimize some metrics:
  - Number of participants included
  - Maximizing earnings
  - Etc.

Goals II

- Real-time (ish)
  - Quick enough so the network doesn’t stall
- Handle a large body of participants

What We Evaluate Against

- Hardware (GPU) accelerated solution
  - SIMD - Single Instruction Multiple Data

Formalism

- *ServiceCall* - unit
  - Request / Offer

- {
  “named-entity-recognition”: offer,
  “part-of-speech-tagger” : request
  }

Formalism

- *AllOf / OneOf* - operators

- ```
  OneOf(
     AllOf({
         “named-entity-recognition”: offer,
         “part-of-speech-tagger”   : request
     }),
     AllOf({
         “named-entity-recognition”: offer,
         “word2vec-embeddings”     : request
     }),
  )
  ```
  ```

Formalism

- Pricing Constraints

- {
  OneOf(
   AllOf({
       “named-entity-recognition”: offer,
       “part-of-speech-tagger”   : request
   }),
   AllOf({
       “named-entity-recognition”: offer,
       “word2vec-embeddings”   : request
   }),
  ),
  Price: >=20
}

Formalism

- We call this a Commitment:
  - OneOf(AllOf(...), AllOf(...)), price: >=20

Scheduler

- Put together a collection of commitments
- The collection satisfies some constraints

Commitment Constraints

- If a commitment is scheduled, exactly one of the OneOf operands must be scheduled

- If a commitment is not scheduled, none of the OneOf operands must be scheduled

AllOf Constraints

- If an AllOf is scheduled, all of the constituent ServiceCalls must be scheduled
- If an AllOf is not scheduled, none of the constituent ServiceCalls must be scheduled

ServiceCall Constraints

- If a ServiceCall request is scheduled, exactly one offer must be scheduled for it
- If a ServiceCall offer is scheduled, at least one request must be scheduled for it

Also,

- Schedule at least one Commitment
  - So the problem becomes unsat
- Additionally 1 variable per ServiceCall p(S) where S is a ServiceCall
- AllOf scheduled in a Commitment:
  - For each ServiceCall in this AllOf:
    - Add +p(S) if ServiceCall is offered
    - Add -p(S) if ServiceCall is requested

Setup

- Pseudobooleans for each Commitment - C_i
- Pseudobooleans for each AllOf - C_i-A_i
- Real valued prices for each ServiceCall - $p(S)$

Setup

- Run encoding through Z3
- Optimize using Z3-opt

Optimize

- # of Commitments scheduled
- maximize ΣC_i

Baseline

- GPU Encoding
- Solves just the matching problem
- Pricing not solved

GPU Kernel

- Encode AllOf
- Exploit “+” (SIMD GPU) to mean scheduling AllOfs
- Regularly prune

AllOf Encoding

Service Call Portion

[0, 1, 0, 1, 1, 0, 0, 0, 1]

Request/Offer
Indicators

Commitment Portion

Take 2 Of These

```
[ 0, 1,  0, 1,  1, 0,   0, 0, 1]
                +
[ 0, 0,  0, 0,  0, 1,   0, 1, 0]
                =
[ 0, 1,  0, 1,  1, 1,   0, 1, 1]
```

Potential Schedule

[0, 0, 0, 0, 0, 0, 0, 0, 0]

Potential Schedule

[0, 0, 0, 0, 0, 0, 0, 0, 0]

+

[0, 1, 0, 1, 1, 0, 0, 0, 1]

=

[0, 1, 0, 1, 1, 0, 0, 0, 1]

Sum = Potential Schedule

- Pool of potential Schedules (frontier)
- Add next AllOf to each
  - New pool of potential Schedules (new frontier)
  - Maintain frontier till all AllOfs are considered
  - Regularly prune

Prune Heuristic

- We are maximizing # of Commitments

Prune Heuristic

[0, 1, 0, 1, 1, 1, 0, 1, 1]

Sum Of These

Thus

- Build encoding
- Set up initial schedule (all zeros - i.e nothing scheduled)
- Pick AllOfs and add to the initial schedule
  - Get a new set of potential schedules
  - Prune the list (can’t double schedule offers etc.)
  - Repeat

Benchmarks

- Pool Of ServiceCalls
- Create Commitment:
    - Sample n
        - N - # of AllOfs in this Commitment
- Create Each AllOf:
    - M - # of ServiceCalls in this AllOf
    - Randomly make them an offer / request
- Randomly sample a price for each Commitment

Hardware

- Z3 Scheduler:
  - Macbook Pro with an Intel(R) Core(TM) i7-5557U CPU @ 3.10GHz with 4 cores
- GPU Kernel:
  - NVidia Tesla K40c graphic card

Key Findings

- Z3 encoding is a lot more compact
- Solves bigger problems
- Substantially faster despite hardware acceleration
- Faster at solving the SAT AND Pricing problems

Abundance / Scarcity Scenario

- Bias the sampling:
  - Abundance : Far more offers than requests
  - Scarcity : Far more requests than offers
  - 70/30 split

Key Finding

- Solver is very fast at dealing with the scarcity case
- Slower in the abundance case
- Potentially takes longer to finish the optimization step when supply is high

Benchmarks + Results

- https://github.com/onai/SMTExperiments

References

- de Moura, L., Bjørner, N.: Z3: An efficient SMT solver. In: Tools and Algorithms for the Con- struction and Analysis of Systems (TACAS) (2008)

- Kovásznai, G., Biro´, C., Erdélyi, B.: Generating optimal scheduling for wireless sensor networks by using optimization modulo theories solvers. In: Proc. Int. Workshop on Satisfiability Modulo Theories (SMT) (2017)

- Kasi, B.K., Sarma, A.: Cassandra: Proactive conflict minimization through optimized task scheduling. In: Proceedings of the 2013 International Conference on Software Engineering. pp. 732–741. ICSE ’13, IEEE Press, Piscataway, NJ, USA (2013)

Questions